

AD A102227

LEVEL II

①

⑥ PERFORMANCE EVALUATION OF MULTIPROCESSOR SYSTEMS  
-----  
CONTAINING SPECIAL PURPOSE PROCESSORS

by

⑩ Jane W.S. / Liu  
Chung A. / Liu

Laung

Department of Computer Science  
University of Illinois  
Urbana, Illinois

and

Nabuyoshi / Miyazaki  
Haruaki / Yamazaki

OKI Electric Industry Company, Ltd.  
Tokyo, Japan

DTIC  
ELECTE  
JUL 30 1981

E  
⑪ 1979

⑫ 33

+ This work was partially supported by the Office of Naval Research  
under Contract No. ONR-N00014-79-C-0775

⑮

DISTRIBUTION STATEMENT A  
Approved for public release;  
Distribution Unlimited

176011

241

81 7 30 058

DTIC FILE COPY

**Best  
Available  
Copy**

Abstract

✓  
The relative merits of two different types of multiprocessor systems are compared in terms of their effective processing capabilities. These two types of multiprocessors are (i) one which contains general purpose processors and (ii) the other which contains special purpose processors. A deterministic model and queueing theoretical models of these systems are described. The potential performance improvement by multitasking is discussed in terms of the number of processors and the degree of concurrency in jobs.

Accession For	
NTIS GRA&I	<input checked="" type="checkbox"/>
DTIC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
<i>in on file</i>	
By	
Distribution/	
Availability Codes	
Dist	Avail and/or Special
A	

## 1. Introduction

In recent years, progress in hardware technology and system architecture has made the design and implementation of large and complex multiprocessor systems possible. By a multiprocessor system, we mean specifically here a computer system which contains two or more closely coupled processors and is in the category of MIMD (Multiple Instruction Streams, Multiple Data Stream) systems. The architectural differences among these systems can be characterized in many different ways [1]. Here, we classify multiprocessor systems according to the types of processors in the system. Some multiprocessor systems consist of identical general purpose processors which share the input job load. Examples of this type of multiprocessor systems include most of well know multiprocessors and closely-coupled computer networks such as IBM 360/65 or 370 MP, Burroughs B 5500 [2], c.m.m.p. [3] and PRIME [4]. Other multiprocessor systems contains special purpose processors (or functionally dedicated) each of which is designed or programmed to perform efficiently a particular type of functions. The types of special purpose processors which have received a great deal of attention in recent years include front-end communication processors designed to deal with input and output of low speed data and line control procedures, back-end processors (or computers) designed to relieve the host system tasks involved in management of data bases, array processors, intelligent graphics terminals, sort-merge processors, etc. designed to perform special functions with speeds normally unachievable using general purpose hardwares. As a matter of fact, one commonly used technique to capitalize the cost/performance potential of VLSI components is to build powerful special purpose processors and use them as attached processors to existing computing systems. Thus, certain functions may be off-loaded for more efficient execution.

Clearly, for a multiprocessor system containing special purpose processors to have comparable cost/performance characteristics, it must have some architectural merits (such as, fast processor speed, more reliable communication paths, etc;) to compensate for the potential lack of such desirable features as fail-softness, expandability, maintainability, etc., provided through redundancy in a system containing general purpose processors.

It is difficult to compare the relative merits of the two types of multiprocessor systems in terms of these criteria in general. In this paper, we are concerned with their relative merits when they are compared in terms of their effective processing capabilities. Using a deterministic model and several approximate queueing theoretical models of multiprocessor systems, their relative performance are compared using various measures of effectiveness.

In Section II, a general deterministic model of multiprocessor systems is described. In this model, each type of special purpose processors is further divided into subtypes with a partial ordering relation defined over the processor subtypes. Thus multiprocessor systems in which some processors are functionally identical but have dedicated memories of different sizes can also be modelled. This model is used to obtain a worst case bound on the performance of priority driven scheduling algorithms.

To study the performance of the two types of multiprocessor systems from another point of view, these systems are modelled using approximate queueing theoretical models in Section III.

A closely related problem is on the potential performance improvement in multiprocessor systems achievable by multitasking. While multitasking system can be effectively modelled with our deterministic model, the queueing models in Section III can only be used for multiprogrammed systems. We discuss in Section IV a special queueing model of multitasking systems and evaluate the potential gain in processing capability achievable by multitasking.

Section V summarizes our conclusions from the results obtained in the previous sections.

## 2. A general deterministic model of multiprocessors systems

Consider a multiprocessor system of  $r$  different types of special purpose processors. Each type of processors is further divided into subtypes. Thus, we shall refer to a processor as a type  $(j,k)$  processor when it is a type  $j$  processor of subtype  $k$ . A partial ordering relation  $<$  is defined over the processor types such that

- (i)  $(j,k)$  and  $(n,v)$  are incomparable<sup>1</sup> if  $j \neq n$ .
- (ii)  $(j,k) < (j,v)$  means that if a task can be executed on a type  $(j,k)$  processor then it can also be executed on a type  $(j,v)$  processor.

A multiprocessor system can then be represented as  $\mathcal{P} = (m_{11}, m_{12}, \dots, m_{1l_1}; m_{21}, m_{22}, \dots, m_{2l_2}; \dots; m_{r1}, m_{r2}, \dots, m_{rl_r}; <)$  where  $m_{jk}$  is the number of type  $(j,k)$  processors and  $<$  is a partial ordering relation over the processor types.

We let :

$$m_j = \sum_{k=1}^{l_j} m_{jk},$$

$$m = \sum_{j=1}^r m_j.$$

For example, the multiprocessor system represented by the directed graph<sup>2</sup> in Figure 1 contains three types of processors. There are three type 1 processors of the same subtype. The 3 processors of type 2 having dedicated memories of different sizes. The execution of a task with certain memory space requirement can take place only on a processor whose memory capacity is larger than or equal to its memory requirement. In our model, these processors are linearly ordered as shown. For the 4 processors of type 3, neither  $(3,1) < (3,2)$  nor  $(3,2) < (3,1)$  but  $(3,3) < (3,1)$  and  $(3,3) < (3,2)$ . This system can be represented as  $\mathcal{P} = (3; 1, 1, 1; 2, 1, 1, <)$ .

<sup>1</sup> Neither  $(j,k) < (n,v)$  nor  $(n,v) < (j,k)$ .

<sup>2</sup> There is an edge from  $(j,k)$  to  $(j,v)$  means  $(j,v) < (j,k)$ .

Let  $\mathcal{T} = \{T_1, T_2, \dots, T_n\}$  be a set of tasks to be executed on a system  $\mathcal{P}$ . A task  $T_i$  is said to be a type  $(j,k)$  task if it can be executed on any type  $(j,v)$  processor for  $(j,k) < (j,v)$  and on no other type of processors. Let  $\alpha$  be a function from  $\mathcal{T}$  to the processor types  $(j,k)$  so that  $\alpha(T_i)$  specifies the type of task  $T_i$ . We denote the time required to complete a task  $T_i$  (on a type  $(j,k)$  processor) by  $\mu(T_i)$  where  $\mu$  is a function from  $\mathcal{T}$  to the reals.  $\mu(T_i)$  shall be referred to as the execution time of  $T_i$ .

We suppose that there is a precedence relation  $<$  defined over the set  $\mathcal{T}$ . That  $T_i$  precedes  $T_j$  (or  $T_j$  follows  $T_i$ ) is written as  $T_i < T_j$  and means that the execution of  $T_j$  cannot begin before the execution of  $T_i$  is completed. A task is said to be executable at a certain time if the tasks preceding it have been completed. Formally, a set of tasks are represented by an ordered quadruple  $(\mathcal{T}, \alpha, \mu, <)$ . We use the notation  $(T_i, \alpha(T_i), \mu(T_i))$  to represent a particular task  $T_i$ .

Consider all type  $(j,k)$  processors for a fixed  $j$ . The smallest subset of types  $\{(j,k_1), (j,k_2), \dots, (j,k_q)\}$  is said to be the dominating set if for any type  $(j,k)$ ,  $(j,k) < (j,k_p)$  for some  $1 \leq p \leq q$ . In other words, any type  $(j,k)$  task can be executed on a processor whose type belongs to the dominating set. For example, in the multiprocessor system shown in Figure 1.  $\{(1,1)\}$ ,  $\{(2,1)\}$  and  $\{(3,1), (3,2)\}$  are dominating sets. We also refer to a type  $(j,k)$  as a maximal type if it is in the dominating set.

For a given dominating set, let

$$m_{j0} = \min \{m_{jk_p} \mid (j,k_p) \text{ is in the dominating set}\}.$$

In other words,  $m_{j0}$  is the minimum of the numbers of processors among all types or processors in the dominating set. Therefore,  $m_{j0}$  is equal to 3, 1, and 1 for  $j=1, 2$ , and 3, respectively, in our example.

We want to determine the performance of a class of scheduling algorithms in which processors are never left idle intentionally. These algorithms are known as priority driven scheduling algorithms and can be described by the priorities assigned to the tasks [5].

Let  $\omega$  and  $\omega'$  denote the completion times of a set of tasks  $(\mathcal{T}, \alpha, \mu, <')$  when executed on a system  $\mathcal{P}$  according to a priority-driven schedule and an arbitrary schedule, respectively. The ratio of  $\omega$  to  $\omega'$  has the following upper bound

$$\frac{\omega}{\omega'} \leq 1 + \sum_{j=1}^r \frac{m_j}{m_{j0}} - \min_{1 \leq j \leq r} \frac{1}{m_{j0}} \quad (1)$$

To prove this inequality, let  $t_j$  denote the total execution of all type  $(j, k)$  tasks for  $k = 1, 2, \dots, l_j$ . For a priority-driven schedule, let  $\phi$  be the total idle time in all processors. Hence the completion time of the priority driven schedule is as given by

$$\omega = \frac{1}{m} \left( \sum_{j=1}^r t_j + \phi \right)$$

Let  $I$  be the sum of lengths of all idle periods during which at least one of each maximal type processors is idle. Let  $s_j$  be the sum of the portion of execution times of all type  $(j, k)$  tasks scheduled during these periods. There is a chain of tasks in  $\mathcal{T}$  such that during these idle periods one of the other processors is executing a task in the chain. Hence

$$I \leq (m-1)\omega'$$

Moreover,

$$\sum_{j=1}^r s_j \geq \frac{I}{m-1} \quad (2)$$

Let  $K_j$  denote the sum of lengths of idle periods during which all the processors of one or more maximal types of the form  $(j, *)$  are busy. We have

$$K_j \leq \frac{m-m_{j0}}{m_{j0}} (t_j - s_j)$$

Combining this inequality with the inequality in (2), we obtain

$$\sum_{j=1}^r K_j \leq \sum_{j=1}^r \frac{m-m_{j0}}{m_{j0}} t_j - \frac{I}{m-1} \left( \min_{1 \leq j \leq r} \frac{m-m_{j0}}{m_{j0}} \right).$$

Since  $t_j/m_j \leq \omega'$  for  $j=1,2,\dots,r$ ,  $\sum_{j=1}^r t_j/m \leq \omega'$ , and

$$\phi \leq I + \sum_{j=1}^r K_j,$$

$$\omega \leq \omega' \left[ 1 + \sum_{j=1}^r \frac{m_j}{m_{j0}} - \frac{1}{m} \sum_{j=1}^r m_j + 1 - \min_{1 \leq j \leq r} \frac{1}{m_{j0}} \right]$$

which reduces to the bound given by Equation (1).  $\square$

When the dominating sets contains only one subtype for all  $j=1,2,\dots,r$ , (that is, there is an unique maximal subtype for all types). The bound given by (1) is the best possible. This fact can be demonstrated by an example which can be found in [6].

For a system containing  $m$  identical general purpose processors, the upper bound in (1) reduces to the well known result [7]

$$\frac{\omega}{\omega'} \leq 2 - \frac{1}{m}.$$

On the other hand, for a job shop problem,  $m_1=m_2=\dots=m_r=1$ . In this case, we have

$$\frac{\omega}{\omega'} \leq r.$$

When the multiprocessor system contains only one type of processors, we have  $m_i=0$  for  $i=2,3,\dots,r$ . The bound given by (1) is simply

$$\frac{\omega}{\omega'} \leq 1 + \frac{m_1}{m_{11}} - \frac{1}{m_{11}}.$$

The bound derived in [8] for processors with different storage capacities is a special case of our result with  $m_{11}=1$ .

### 3. Queueing models of multiprocessor systems

To model a multiprocessor system probabilistically, we assume that the arrival process of jobs requesting service of the system is Poisson with parameter  $\lambda$ . Each job may be decomposed into a number of different tasks. There are altogether  $r$  different types of tasks. (For example, consider a system in which some jobs are decomposed into an input task followed by compilation, computation and output tasks while the other jobs are decomposed into input, sorting and merging, and output tasks. In this case, there are 5 different types of tasks). We say that these tasks are generated by the job. Let  $N_i$  denote the number of tasks of type  $i$  and  $N$  be the total number of tasks generated by a job. We assume that  $N_i$ 's are statistically independent random variables.

It is sufficient to consider the relative speeds of the processors. We choose to measure the speeds of special purpose processors with respect to the speed of a general purpose processor. In particular, we call the relative speed of a special purpose processor with respect to a general purpose processor the capacity of the special purpose processor. For example, if a task takes  $1/\mu$  units of time to be completed by a general purpose processor, then it takes  $(1/\mu)(1/C)$  units of time to be completed by a processor with capacity  $C$ .

We refer to the time required to complete a task in a system as the execution time (or service time) of the task in that system. In particular, the execution time of the task on a general purpose processor is called the amount of work for that task. Hence, if a special purpose processor with capacity  $C$  completes the given task within  $t$  sec, then the amount of work for that task is  $t \cdot C$  units.

We measure the effectiveness of a multiprocessor system by the average total amount of work remaining in the system in statistical equilibrium. That is, the total time required for a general purpose processor to complete all tasks being served and waiting for service in the system. This performance measure is chosen since it does not depend the queueing discipline used to schedule the tasks in the system.

### 3.1. Systems with independent input processes

When the tasks generated by jobs are independent<sup>3</sup>, we approximate the arrival processes of different types of tasks by independent Poisson processes<sup>4</sup>. Let  $\lambda_i$  denote the average arrival rate of tasks of type  $i$ . A multiprocessor system containing  $m$  general purpose processors can be approximately modelled by the M/G/m queue shown in Figure 2a. In this case, when all processes are busy, the tasks joins a common queue (for example, as in B 5500 [2]). Similarly, a multiprocessor system containing  $r$  types of special purpose processors can be modelled by the multiserver system in Figure 2b. Let  $m_i$  denote the number of type  $i$  processors (i.e., those designed to execute tasks of type  $i$  only). These  $m_i$  processors are referred to collectively as the  $i$ th subsystem. A type  $i$  tasks joins the  $i$ th queue upon its arrival.

It is difficult to analyze the general behavior of multiserver queues since service times in our case are non-exponential. We consider here several special cases :

#### 3.1.1. Systems with one processor of each type

For the case where  $m_1=m_2=\dots=m_r=m=1$ , expressions for average total amount of work remaining in both types of systems can be obtained easily [9]. When the amount of work for type  $i$  tasks is exponentially distributed with parameter  $\mu_i$  ( $i=1,2,\dots,r$ ), the average total amount of work in the  $i$ th subsystem with one special purpose processor of capacity  $C_i$  is

$$\bar{W}_{si} = \frac{1}{\mu_i} \frac{\rho_i}{1-\rho_i}$$

where  $\rho_i = \frac{\lambda_i}{C_i \mu_i}$ . Hence the average total amount of work remaining in a system containing  $r$  special purpose processors is given by

<sup>3)</sup> In the sense that they can be executed simultaneously.

<sup>4)</sup> Multiprocessor systems containing identical processors may be modelled more accurately by the M/M/m bulk arrival queueing system discussed in Section IV.

$$\bar{W}_S = \sum_{i=1}^r \frac{1}{\mu_i} \frac{\rho_i}{1-\rho_i} \quad (3)$$

Let  $\bar{W}_g$  be the average total amount of work remaining in a system containing a general purpose processor. Since the average execution time of all tasks is equal to

$$\frac{1}{\lambda'} \sum_{i=1}^r \lambda_i / \mu_i \quad \text{and} \quad \lambda' = \sum_{i=1}^r \lambda_i$$

we have

$$\bar{W}_g = \frac{1}{1-\rho} \sum_{i=1}^r \frac{\lambda_i}{\mu_i} \quad (4)$$

$$\text{where } \rho = \sum_{i=1}^r \lambda_i / \mu_i.$$

To compare the performance of the two systems, we assume that the capacities,  $C_i$ , of the special purpose processors are chosen to minimize,  $\bar{W}_S$ , subject to the constraint  $\sum_{i=1}^r C_i = C$ . The values of  $C_i$  that minimize  $\bar{W}_S$  are given by

$$C_i = \frac{\lambda_i}{\mu_i} + (C-\rho) \frac{\sqrt{\lambda_i}}{\mu_i} \left( \sum_{j=1}^r \sqrt{\lambda_j} / \mu_j \right)^{-1}$$

For these values of  $C_i$ ,  $\bar{W}_S$  in Equation (3) becomes

$$\bar{W}_{so} = \frac{1}{C-\rho} \sum_{i=1}^r \left( \frac{\sqrt{\lambda_i}}{\mu_i} \right)^2$$

Moreover,  $\bar{W}_{so} \leq \bar{W}_g$  as long as the total capacity of all special purpose processors,  $C$ , is such that

$$C \geq (1-\rho) \left( \sum_{i=1}^r \sqrt{\lambda_i} / \mu_i \right)^2 / \left( \sum_{i=1}^r \lambda_i / \mu_i^2 \right) + \rho$$

This result indicates that in order for a system containing special purpose processors to be equally effective as a system containing general processors, the special purpose processors must be made sufficiently fast.

### 3.1.2. Systems with arbitrary number of processors

To compare the effectiveness of systems containing arbitrary number of processors of each type, we consider two analytically tractable cases :

(i) Deterministic task execution time

In particular, the amounts of work required for all tasks are constant and identical. Without loss of generality, let the amount of work required by a task be one unit of time or a time slot. If the average interarrival time of jobs is sufficiently long compared with this time slot, then we can approximate the exponential distribution of interarrival time by a geometric distribution. We assume that the job scheduler assigns tasks to the processors at the beginning of each time slot. If  $t_n$  is the  $n$ th epoch of this time slot, the total number of tasks at  $t_n + 0^+$  forms a Markov chain (and so is the total amount of remaining work in the system at  $t_n + 0^+$ ).

Let  $\Pi_i$  be the probability of having  $i$  tasks in a system containing  $m$  general purpose processors at equilibrium. It has been shown that when the number of processors is larger than  $\lambda E(N)$ , the average number of tasks in the system is given by [10].

$$\bar{n} = \sum_{i=0}^{m-2} \left( \frac{1}{1-z_i} \right) - \frac{m(m-1)}{2(m-A'(1))} + A'(1) + \frac{A''(1)}{2(m-A'(1))},$$

where  $A(z) = \sum_{i=0}^r A_i(z)$  is the generating function of the random variable  $N$  and  $A_i(z)$  is the generating function of  $N_i$ .  $z_0, z_1, \dots, z_{m-2}$  are the zeros of  $(1 - \frac{z^m}{A(z)})$  within the unit circle.

In a multiprocessor system containing  $r$  types of special purpose processors, the execution time of a type  $i$  task is  $1/C_i$ . Let  $B_i(z)$  be the generating function of the number of tasks that arrive during the execution of a type  $i$  task. Clearly,

$$B_i(z) = 1 - \frac{\lambda}{C_i} + \frac{\lambda}{C_i} A_i(z).$$

The average number of task in the  $i$ th subsystem,  $\bar{n}_i$ , is given by

$$\bar{n}_i = \sum_{j=0}^{m_i-2} \frac{1}{1-z_j^{(i)}} - \frac{m_i(m_i-1)}{2(m_i-B_i'(1))} + \frac{B_i''(1)}{2(m_i-B_i'(1))}.$$

where  $z_j^{(i)}$  are zeros of  $(1 - z^{m_i}/A_i(z))$  within the unit circle.

Unfortunately, it is difficult to obtain the general behaviors of  $\bar{W}_g$  and  $\bar{W}_s$  except for the case where  $A(z)$  and  $A_i(z)$  are polynomials of  $z^m$  and  $z^{m_i}$ , respectively.

(ii) Tasks with exponential execution time in systems under heavy traffic

We assume here that the amount of work,  $S_i$ , required for type  $i$  tasks is exponentially distributed with parameters  $\mu_i$ . Let  $S = S_1 + S_2 + \dots + S_r$  be the total amount of work of a job. Let us denote by  $t_n$  the epoch when the  $n^{\text{th}}$  job arrives. Let  $W_n$  be the amount of work remaining at  $t_n$  in a system containing  $m$  general purpose processors,  $A_n$  be the amount of work completed in the duration  $[t_{n-1}, t_n]$ , and  $B_n$  be the amount of work arriving at  $t_n$ . Then  $W_{n+1}$  can be expressed as

$$W_{n+1} = \begin{cases} W_n + B_n - A_n & \text{if } W_n + B_n - A_n \geq 0 \\ 0 & \text{if } W_n + B_n - A_n < 0 \end{cases} \quad (5)$$

Under heavy traffic conditions, all  $m$  processors are busy during  $[t_{n-1}, t_n]$ . Therefore,  $A_n$  has the same distribution as  $mA$  where  $A$  is the interarrival time between two jobs.

$B_n$  has the same distribution as  $S$ . Hence we can write Equation (5) as :

$$W_{n+1} = \begin{cases} W_n + (S - mA) & \text{if } W_n + S - mA \geq 0 \\ 0 & \text{if } W_n + S - mA < 0 \end{cases}$$

However, the expression in the right hand side of this equation is the waiting time of the  $(n+1)^{\text{th}}$  job in a system which consists of one processor with the execution time of a job being  $S$  and the average interarrival time between jobs being  $mA$ . Since

$$E[mA] = \frac{m}{\lambda}$$

$$E[S] = \sum_{i=1}^r \frac{1}{\mu_i}$$

$$E[S^2] = \left( \sum_{i=1}^r \frac{1}{\mu_i} \right)^2 + \sum_{i=1}^r \left( \frac{1}{\mu_i} \right)^2$$

The mean waiting time in queue is

$$\frac{1}{2}(\lambda/m) E(S^2) \left(\frac{1}{1-\rho}\right)$$

where

$$\rho = (\lambda/m) \sum_{i=1}^r (1/\mu_i)$$

Moreover, the average total amount of work remaining,  $\bar{W}_g$ , is

$$\bar{W}_g = \frac{1}{2} \frac{\lambda}{m} \frac{1}{1-\rho} \left[ \sum_{i=1}^r \left(\frac{1}{\mu_i}\right)^2 + \sum_{i=1}^r \frac{1}{\mu_i} \right] \quad (6)$$

Similarly, in the system contain  $r$  types of special purpose processors under heavy traffic conditions, the average total amount of work remaining in the  $i^{\text{th}}$  subsystem,  $\bar{W}_{Si}$ , is equal to the average waiting time in queue in a single processor system with interarrival time of jobs being  $C_i m_i A$  and the execution time for a job being  $S_i$ . Since  $S_i$  and  $C_i m_i A$  are exponentially distributed, we have

$$\bar{W}_{Si} = \frac{\frac{\lambda}{C_i m_i} \frac{2}{\mu_i^2}}{2} \frac{1}{1-\rho_i}$$

where

$$\rho_i = (\lambda/C_i m_i) \left(\frac{1}{\mu_i}\right)$$

Thus the average total amount of work remaining,  $\bar{W}_S$ , is

$$\bar{W}_S = \sum_{i=1}^r \bar{W}_{Si} = \lambda \sum_{i=1}^r \left(\frac{1}{\mu_i}\right) \left(\frac{1}{\mu_i C_i m_i - \lambda}\right) \quad (7)$$

We can minimize the value of  $\bar{W}_S$  with respect to  $C_i m_i$  under the condition  $Cm = \sum_{i=1}^r C_i m_i$ . It can be shown [10] that the values of  $C_i m_i$  that minimizes  $\bar{W}_S$  is given by

$$C_i m_i = \frac{C\lambda}{\mu_i}$$

Furthermore, the minimum value of  $\bar{W}_S$  is

$$\bar{W}_{So} = \sum_{i=1}^r \frac{1}{\mu_i} \frac{\rho}{C-\rho}$$

That is

$$\bar{W}_{So} = \frac{m}{\lambda} \frac{\sigma^2}{C-p}$$

Figure 3 shows the curves  $\bar{W}_{So}$  and  $\bar{W}_g$  in the case of  $r=3$ ,  $(\frac{1}{\mu_i}) = i+1$  for  $i=0,1,2$ ,  $C_i=C$  and  $m=10$ . Under heavily loaded conditions, the optimized system with special purpose processors behaves better. Even in the case  $C=1$  (i.e., no improvement of processor speed), the value of  $\bar{W}_{So}$  is almost comparable to that of  $\bar{W}_g$ . This result is an expected one since the flexibility in scheduling does not make much difference under heavily loaded conditions.

Figure 4 describes another case where  $r = m = 3$ ,  $(\frac{1}{\mu_i}) = (0.9)^i$ ,  $C_i = C$  for  $i=0,1,2$ . Therefore,  $m_i=1$  for all  $i$ . Since all subsystems consist of a single processor,  $\bar{W}_g$  is not optimized in this case.

### 3.2. Queueing network model

There are two methods to schedule jobs in a multiprocessor system : (i) multiprogramming and (ii) multitasking. By multiprogramming, one usually means that the system may execute more than one job simultaneously. By multitasking, one means that the system may execute tasks in a job concurrently if the job can be divided into independently executable tasks. Multiprocessor systems without multitasking can be modelled using queueing networks. We model a multiprocessor systems containing special purpose processors approximately as an open network of Jackson servers. Let  $R_{ij}$  be the probability of a task of type  $j$  is to be executed after the completion of a task of type  $i$ . Let  $f_i$  be the probability that the first task to be performed for a job is of type  $i$ , then the arrival rate,  $\lambda_i$ , of tasks of type  $i$ ,  $i=1,2,\dots,N$  are given by the set of equations

$$\lambda_i = \lambda f_i + \sum_{j=1}^r \lambda_j R_{ji}$$

According to Jackson's theorem when  $\lambda_i / m_i C_i \mu_i < 1$  for all  $i$ , the equilibrium probability of finding  $n_i$  tasks in subsystem  $i$  is given by

$$p_1(n_1) p_2(n_2) \dots p_r(n_r)$$

where  $p_i(n_i)$  is the equilibrium probability of finding  $n_i$  tasks in an M/M/ $m_i$  queue with input rate  $\lambda_i$  and average execution time  $1/\mu_i C_i$ . Hence, for a given set of  $\lambda_i$ , our results described above are still valid here.

A special case of interest is one where each job generates  $r$  different types of tasks that must be executed in fixed sequence as in a pipelined system. In this case, a system consisting of special purpose processors can be modelled by a series of  $r$  M/M/ $m_i$  queues. With the average execution time of a job on a processor in the  $i$ th subsystem being  $1/\mu_i C_i$ , the average amount of work remaining in the subsystem containing  $m_i$  type  $i$  processors is

$$\bar{W}'_{Si} = \frac{\rho_i}{1-\rho_i} \frac{1}{\mu_i}$$

where  $\rho_i = \lambda/C_i m_i \mu_i$ . Hence the average amount of work remaining for all jobs in the  $i$ th subsystem is

$$\bar{W}_{Si} = \bar{n}_i \sum_{j>i} \frac{1}{\mu_j} + \bar{W}'_{Si}$$

where  $\bar{n}_i$  is the average number of jobs in the  $i$ th system. Since  $\bar{n}_i = \rho_i / (1-\rho_i)$ , we have

$$\bar{W}_{Si} = \frac{\rho_i}{1-\rho_i} \sum_{j=i}^r \frac{1}{\mu_j}$$

The average total number of work remaining in the pipelined multiprocessor system is

$$\bar{W}_S = \sum_{i=1}^r \sum_{j=i}^r \frac{\lambda}{C_i m_i \mu_i - \lambda} \frac{1}{\mu_j}$$

Note that in a system with  $m$  general purpose processors, the processors are not connected in pipeline. Hence the expressions for  $\bar{W}_g$  in Equation (6) is valid here also in heavy loading condition.

Figure 5 describes the behavior of  $\bar{W}_g$  and  $\bar{W}_S$  in the case of  $r = m = 3$ ,  $C_i = c$  and  $(\frac{1}{\mu_i}) = (0.9)^i$ ,  $i=0,1,2$ .

#### 4. Performance of multiprocessors systems with multitasking

Generally, jobs may contain independent and thus concurrently executable tasks. Both turn around time and system resource utilization in multiprocessor system may often be improved by allowing processors to execute in parallel tasks identified either by programmers of the jobs or by the compiler system to be concurrently executable. In this section, we want to determine the potential performance improvement by multitasking. Unfortunately, the models used in Section II and III are deficient in one way or another for this purpose. In the deterministic model, the structures within an individual job are effectively described in the general model of a task set. However, all jobs have the same structure and, more restrictively, are assumed to arrive for service at the same time. On the other hand, in the queueing models used in Section III, the issue of task synchronization is completely ignored. The model described in Figure 2 can be used only in the case where jobs consist of independent tasks while queueing network models of multiprocessor systems can be used only for systems without multitasking.

When the degree of concurrency is small ( $=2$ ), the model in [11] can be used. However, the case with degree of concurrency being two is not an interesting one since it has been shown that multitasking is not a good way to make effective use of multiprocessor system resources when the number of processors is small. This result is due to the fact that negative of larger overhead. On the other hand, it is said that multitasking is essential for a multiprocessor system containing a large number of processors [12].

We study here the dependency of potential improvement achievable by multitasking on the degree of concurrency and number of processors. For this purpose, we discuss first the type of job structures considered here and then a model of multiprocessor systems with multitasking.

##### 4.1. Job structures

Similar to our deterministic model, the structures of jobs can be described by a probabilistic model represented graphically as shown in Figure 6a. In this graph, '—' represents a task, '{' represents the task

generation (e.g. by statements such as fork, cobegin, etc.), and '}' represents the task synchronization (identified by statements such as join, coend etc.). A job consists of many stages of tasks. A stage is either a set of tasks in the same column between a pair of '{' and '}', or a task if it is not immediately preceded by '{'. A stage consists of a random number of tasks, and service times of tasks in each stage are stochastically independent. Tasks in the same stage can be executed independently. Tasks in an inner stage (i.e., tasks in inner brackets) are considered as a part of an outer stage. The execution of tasks in the stage to the left begins before the tasks in stages to its right. It is difficult to model the topological structure of the job, i.e., relations between stages. Therefore, we consider here two approximate models of the job structure : (1) no synchronization model shown in Figure 6b, and (2) full synchronization model shown in Figure 6c. In no synchronization model, we assume that there is no task synchronization. A task simply disappears after being served with certain probability, or it is followed by another stage. In full synchronization model, tasks may not generate new tasks; they must be synchronized immediately after their completions. Thus there are no inner stages in a stage. No synchronization model preserves the topological structure of the job to a certain degree, and can represent complex job structures. However, it is difficult to keep track of a job ; one cannot use it for the purpose of evaluation of job turn around time. Full synchronization model reduces the topological structure into simple linear structure and is easy to analyse. It is the model of job structure used in the following.

#### 4.2. Queueing model of multiprocessor systems with multitasking

We use open queueing networks such as the one shown in Figure 7 to model multiprocessor systems with multitasking. The network has only one external source and one sink. Each node in the network consists of a number of processors and serves a stage of tasks in a job. These processors may be of different types if the stage which it serves consists of several classes of tasks. After the arrival of a job to a node, tasks in the stage are served concurrently whenever possible. The service of a job in a node is completed when services of all tasks in the same stage are completed. The completed job proceeds to a next node (or leave the network) determined randomly with given probabilities. Thus a job may be repeatedly served by

a node. The number of tasks in a stage may vary from node to node.

In general, it is impossible to find the probability distributions of number of jobs in each node in such queueing network. Again, we consider here only one special case.

#### 4.3. A model for multitasking systems containing identical processors

Multiprocessor systems with identical general purpose processors can be approximately modelled using the single node model in Figure 8. Again, we assume that the job arrival is a Poisson process with the average arrival rate  $\lambda$ . Each job consists of  $N$  independent tasks with identically distributed service times. The distribution of  $N$  is arbitrary but has finite first and second moments.  $N$  is referred to the degree of concurrency in the job. The service times of tasks are statistically independent, exponentially distributed with mean  $1/\mu$ . The job service time is the sum of task service times, i.e.,  $S_j = S_1 + S_2 + \dots + S_N$ . Thus the model of multiprocessor systems is specified by a 4-tuple  $(\lambda, \mu, m, N)$ . It is a bulk arrival M/M/m queueing system.

##### Number of tasks in system

Let  $a_i$  be the probability that  $N$  is equal to  $i$  for  $i=0,1,2,\dots$ . ( $a_0=0$  and  $a_i \geq 0$  for  $i \neq 0$ ) and  $A(z)$  the generating function of  $\{a_i\}$ . Let  $K$  be the number of tasks in M/M/m bulk arrival queueing system.  $K$  is a Markov chain. Its stationary distribution  $\{p_k\}$ , if exists, is given by [13]

$$P(z) = \frac{(1-z)\mu \sum_{k=0}^{m-1} (m-k)p_k z^k}{m\mu(1-z) - \lambda z(1-A(z))},$$

where  $\mu_k = \min(k\mu, m\mu)$ . Let  $\rho = \lambda/\mu$  and  $u = \lambda E(N)/m\mu$ . Because  $p_k$  is a linear function of  $p_0, p_1, p_2, \dots, p_{k-1}$ , we write  $p_k = f_k p_0$  for  $k \leq m$ . Hence

$$P(z) = \frac{m(1-u)(1-z) \sum_{k=0}^{m-1} (m-k)f_k z^k}{[m(1-z) - \rho z(1-A(z))] \sum_{k=0}^{m-1} (m-k)f_k} \quad m < \infty.$$

The condition under which the stationary distribution exists is  $u = \lambda E(N)/\mu < 1$ . It is easy to see that  $u$  corresponds to the utilization factor of the processors. For  $m = \infty$

$$P(z) = \exp\{-\rho \int_z^1 [(1-A(t))/(1-t)] dt\}$$

Thus, the average number of tasks in the system is

$$E(K) = \frac{u(1+E(N^2)/E(N))}{2(1-u)} + \frac{\sum_{k=1}^{m-1} (m-k)k\bar{f}_k}{\sum_{k=0}^{m-1} (m-k)\bar{f}_k} \quad (8)$$

For  $m < \infty$ , for  $m = \infty$ ,  $E(K) = \rho E(N)$ .

#### Task waiting time

To find the task waiting time for FCFO (First Come First Serve) discipline, we define the virtual task waiting time as the length of time a task spends in the system (or in the queue) if it arrives at random instant. Then it can be shown that the distribution of the task waiting time is the same as the distribution of the virtual task waiting time [13]. Thus we can get the task waiting time by calculating the virtual task waiting time.

Let  $W_j$  be the waiting time (in the queue) of the  $j$ -th task in a job when we number tasks in the order of them being served. If there are  $k$  tasks in the system just before its arrival,

$$E(W_j | K=k) = \begin{cases} 0 & k+j \leq m \\ (k+j-m)/\mu & k+j > m \end{cases}$$

The above expression is obtained because all processors are busy while the  $j$ -th task (in the arrived job) is waiting. Moreover, the average inter-departure time is equal to the minimum of the service time of  $m$  servers, i.e.,  $1/\mu$  when all servers are busy. The unconditional average is,

$$E(W_j) = \frac{1}{\mu} \left[ E(K) + (j-m) + \frac{m(1-u) \sum_{k=0}^{m-j-1} (m-j-k)f_k}{\sum_{k=0}^{m-1} (m-k)f_k} \right]$$

Let  $W$  be the waiting time in the system of a task and  $W_q$  be the waiting time in the queue of a task. Then,

$$\Pr(W_q > t) = \sum_{j=1}^{\infty} a_j \sum_{i=1}^j \Pr(W_i > t)/E(N).$$

Hence,

$$E(W) = \sum_{i=1}^{\infty} a_i \sum_{j=1}^i E(W_j)/E(N) + 1/\mu$$

### The job waiting time

Let  $V$  be the waiting time in system for a job, i.e., the time between the arrival and the departure of a job. Here the departure of a job means the departure of the last task in a job to leave the system. Let  $K_r$  be the number of tasks in a job still remaining in the system when service for the last task in the job begins. (Notice that the last task to be served is not necessarily the last task to leave the system). Let  $S_r$  be the time required to complete the service of all remaining tasks in the job after the service of the last task begins. The conditional average of  $S_r$  is,

$$\begin{aligned} E(S_r | K_r = n) &= E(\max(S_1, S_2, \dots, S_n)). \\ &= (1/\mu) \sum_{i=1}^n 1/i \end{aligned}$$

Because  $V = W_2 + S_r$  where  $W_2$  is the waiting time in queue of the last task,

$$E(V) = E(W_2) + (1/\mu) \sum_{n=1}^m \Pr(K_r = n) \sum_{i=1}^n 1/i$$

It is possible to compute  $E(V)$  as done in [13]. However, the computation of its numerical values is not efficient. The following bounds of  $E(V)$  are more useful. Because  $E(S_r | K_r = n)$  is an increasing function of  $n$  and clearly  $\Pr(K_r = n) = 0$  for  $n=0$  and  $n > \min(N, m)$ , we have the lower bound

$$E(V) \geq E(W_2) + 1/\mu$$

and the upper bound

$$E(V) \leq E(W_2) + (1/\mu) \sum_{j=1}^{\infty} a_j \sum_{k=1}^{\min(j,m)} 1/k$$

Together, these bounds give us a good approximation of  $E(V)$ . In order to emphasize the fact that  $E(V)$  is a function on  $m$ , we write it as  $E_m(V)$ .

#### 4.4. Performance improvement with multitasking

The queueing system may be adopted to model multiprocessor systems with or without multitasking in addition to uniprocessor systems with the same capacity as the multiprocessors (i.e., it is  $m$  times faster) by adjusting the 4 parameters. To do so, we note first that the performance of multiprocessor systems may be decreased by memory interference and overhead. These effects can be taken into account by assuming that the service time for a job increases in multiprocessor systems. Let  $S$  be the service time for a job in uniprocessor systems. We assume that  $aS$  and  $bS$  are the service time for a job in multiprocessor systems with and without multitasking respectively. The values of  $a$  and  $b$  depend on  $m$ ,  $N$  and  $\lambda$ . Generally,  $a \geq b \geq 1$ .

In a processor system with multitasking, a job departs only when all tasks are completed. The 4-tuples which specifies the model for the system is  $(\lambda, \mu E(N), m, N)$ . In a multiprocessor system with multiprogramming but not multitasking, all tasks in a job are executed in sequence. Hence only one processor is assigned to a job. But many jobs may be executed simultaneously. In this case, the 4-tuple is  $(\lambda, \mu/b, m, 1)$ . Similarly, for a uniprocessor system the 4-tuple is  $(\lambda, m\mu, 1, 1)$  since the processor is  $m$  times faster. All equations in this section are expressed in terms of  $(\lambda, \mu, m, N)$ , where parameters should be substituted by different parameters for different systems. Moreover, since  $E_m(K)$  should be interpreted for different systems, (i.e., it is the average number of tasks for the system with multitasking and is the average number of jobs for the system without multitasking) we choose again to use the total amount of work remaining  $E_m(R)$ , instead of  $E_m(K)$  as a criterion of comparison ( $E_m(R)$  is defined as the required time to complete all remaining tasks (or jobs) in the system by a unit speed processor). Note that  $E_m(R)$  is independent on service disciplines (priority, etc.) of systems although  $E_m(K)$  is not.  $E_m(R)$  is obtained as follows : (i) for a multiprocessor system with multitasking

$$E_m(R) = aE_m(K)/\mu E(N)$$

where  $E_m(K)$  is obtained from Equation (8) by substituting  $(\lambda, \mu, m, N)$  by  $(\lambda, \mu E(N)/a, m, N)$ , (ii) for a multiprocessor system without multitasking

$$E_m(R) = bE_m(K)/\mu$$

where  $E_m(K)$  is obtained from Equation (8) for the 4-tuple  $(\lambda, \mu/b, m, 1)$  (iii) for a multiprocessor system

$$E_m(R) = E_m(K)/\mu$$

where  $E_m(K)$  is obtained from Equation (8) for the 4-tuple  $(\lambda, \mu, 1, 1)$

The values of parameters  $a$  and  $b$  can be determined based on the study of memory interference and overhead. Here, we assume the ideal case, i.e.,  $a = b = 1$ , in the following comparison. From Figure 9, we note that the performance of the uniprocessor system is the best and the performance of the multiprocessor system without multitasking is the worst consistency for all  $E(N)$ . Their difference is larger for larger value of  $m$ . The performance of the multiprocessor system with multitasking improves for larger  $E(N)$ . Indeed, the performance of the multiprocessor system with multitasking is almost same as the performance of the uniprocessor if  $E(N) \geq m$ . The effect of the number of processors on the performance of the systems is shown in Figure 10. The traffic intensity plotted is proportional to the number of processors.  $E_m(R)$  increases as  $m$  increases although  $E_m(R)/m$  decreases, and the increasing rate is larger for smaller  $E(N)$ . On the other hand,  $E_m(V)$  decreases as  $m$  increases and the decreasing rate is larger for larger  $E(N)$ . If we assume that traffic intensity is constant, both performance measures are improved in all systems as  $m$  increases. However, the difference of performance between these systems is smaller for larger  $u$  (system utilization factor).

## 5. Summary

It is our objective here to evaluate the merits of using special purpose processors in multiprocessor systems. For this purpose, we propose several models of multiprocessor systems and use them to obtain different performance measures which may be used as criteria for comparison of the processing capabilities of the two types of multiprocessor systems.

The general deterministic model of multiprocessor system described in Section II include as special cases many models (e.g. models of system with identical processors, processors of different size memories and different processors in job shop problems) used in previous studies. We may conclude from the result in Section II that according to a priority driven schedule, the completion time of a set of tasks executed on a system containing  $r$  types of processors can be very poor for large  $r$ . The relative inferior performance of multiprocessor systems containing special purpose processors is clearly due to the loss in scheduling flexibility in such systems. When such a system is used in real-time environments, scheduling algorithms with better worst case behavior than arbitrary priority driven schedule need to be found.

To determine the minimum ratio between the speeds of special purpose processors and general purpose processors to achieve the same overall system capabilities, several approximate queueing models are proposed. Using the total amount of remaining work in system as a basis of comparison, the two types of multiprocessor systems are compared quantitatively for different speeds of the special purpose processors in the case when the systems are multiprogrammed (but are not multitasked).

It is different to model multiprocessor systems with multitasking in general. We discuss a probabilistic representation of job structures and a general queueing network model for systems with multitasking. The model may be used in simulation studies but is, unfortunately, analytically untractable. We purpose here to use a M/M/m queueing system with bulk arrival as an approximate model of systems with multitasking. The performance of multiprocessor systems with and without multitasking are compared with an

uniprocessor system with equal capability. Our results confirm that multitasking improves the performance of multiprocessor system when the degree of concurrency in jobs is large enough.

# REFERENCES

-----

- [ 1] Enslow P.H. Jr., "Multiprocessor Organization - a survey", ACM Comp. Surveys, Vol. 9, n° 1, March 1977.
- [ 2] Davis R.L., S. Zucker, C.M. Campbell, "A Building Block Approach to Multiprocessing", Proc. of Spring Joint Computer Conference, pp. 685-703, 1972.
- [ 3] Wulf W.A., C.G. Bell, "C.mmp - A multi-miniprocessor", Proc. of Fall Joint Computer Conference, pp. 765-777, 1972.
- [ 4] Baskin H.B., B.R. Borgerson, R. Roberts, "PRIME- A modular Architecture for Terminal Oriented Systems", Proc. of Spring Joint Computer Conf., Vol. 40, pp. 431-437, 1972.
- [ 5] Coffman E.G.Jr., P. Denning, "Theory of Operating Systems", John Wiley and Sons, 1975.
- [ 6] Liu J.W.S, C.L. Liu, "Performance analysis of multiprocessor systems containing functionally dedicated processor", Depart. of computer Science, University of Illinois at Urbana, Champaign, TR UTUCDCS-R-835, 1977.
- [ 7] Graham R.L., "Bounds on multiprocessing timing anomalies", SIAM J. Applied Math., Vol. 17, N° 2, 1969.
- [ 8] Kafura D.G., V.Y. Shen, "Scheduling Independent Processors with Different Storage Capabilities", Proc. ACM National Conference, pp. 161-166, 1974.
- [ 9] Kleinrock L., "Queueing Systems", Vol. 1, Theory, John Wiley and Sons, 1975.
- [10] Yamazaki H., "Performance Evaluation of multiprocessor Systems", Tech. report UTUCDCSR-77-891, Department of Computer Science, University of Illinois at Urbana - Champaign.
- [11] Sauer C.H. and K.M. Chandy, "The Impact of Distribution and Disciplines on Multiple Processors Systems", Comm. of ACM, vol. 22, N° 1, Jan. 1979.
- [12] Fuller S.M., J.K. Custerhout, L. Raskin, P et al., "Multi-microprocessors An Overview and Working Example", Proc. of IEEE, Vol. 61, n° 2, Feb. 1978.
- [13] Miyazaki N., "A performance analysis of multiprocessor systems with multitasking", Ms Thesis, Department of Computer Science, University of Illinois at Urbana, Champaign, 1979.

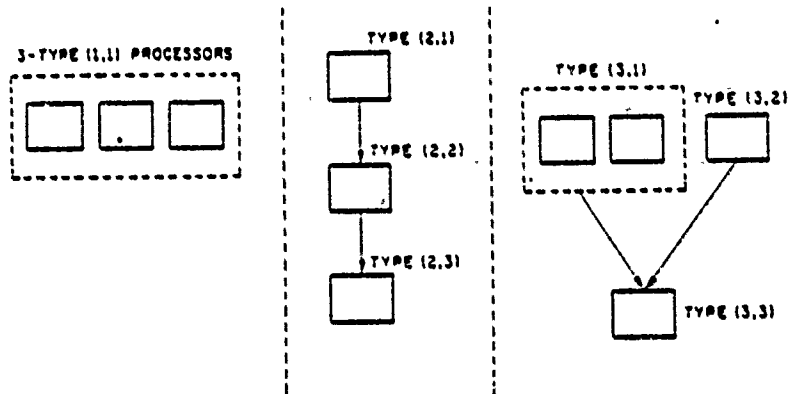


Figure 1. An Example of multiprocessor systems containing special purpose processors

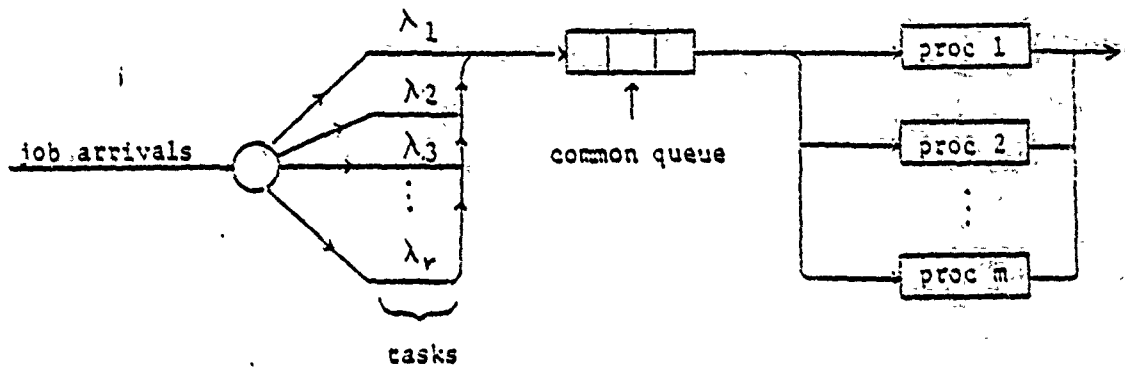
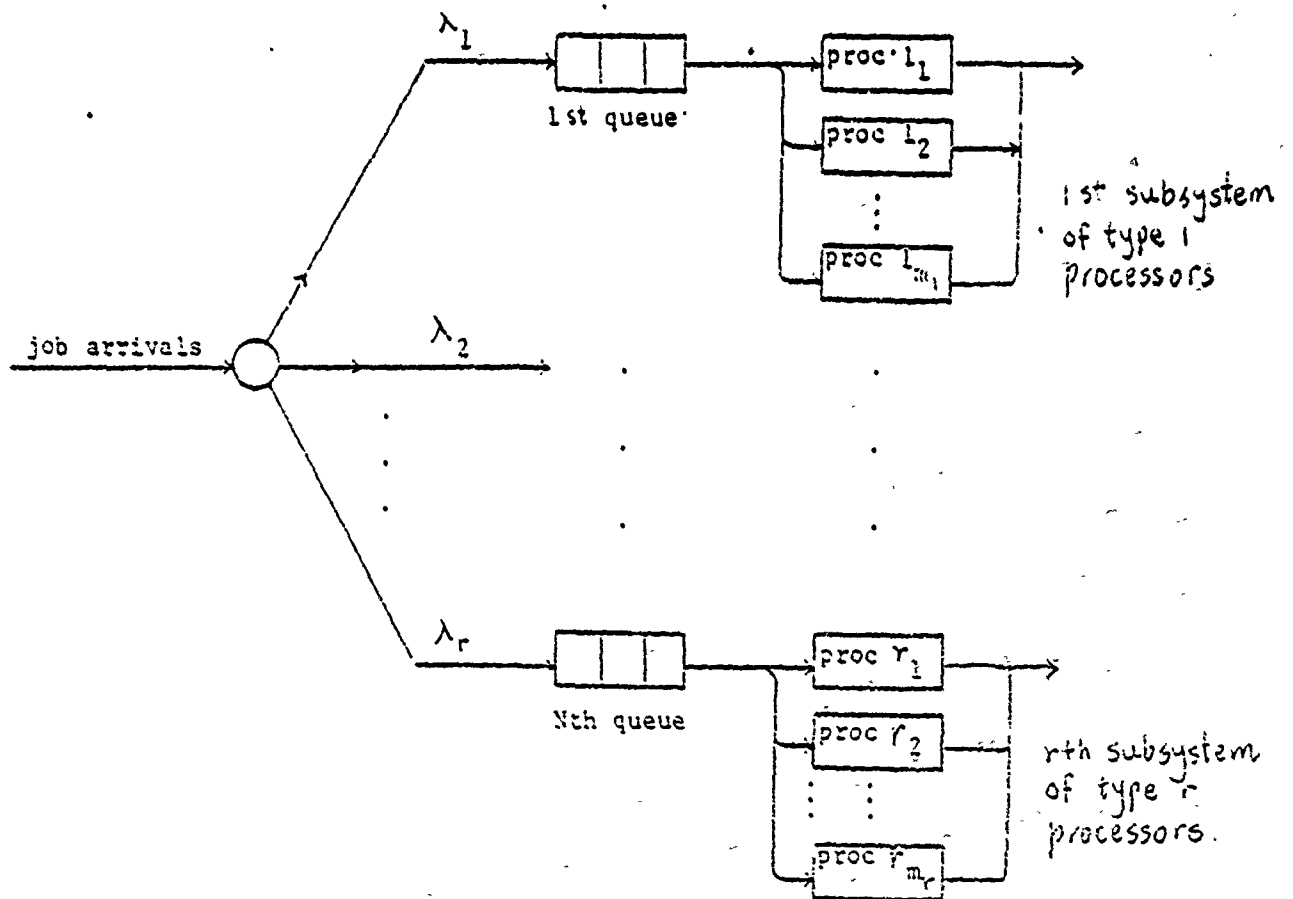


Figure 2a. A system with general purpose processors



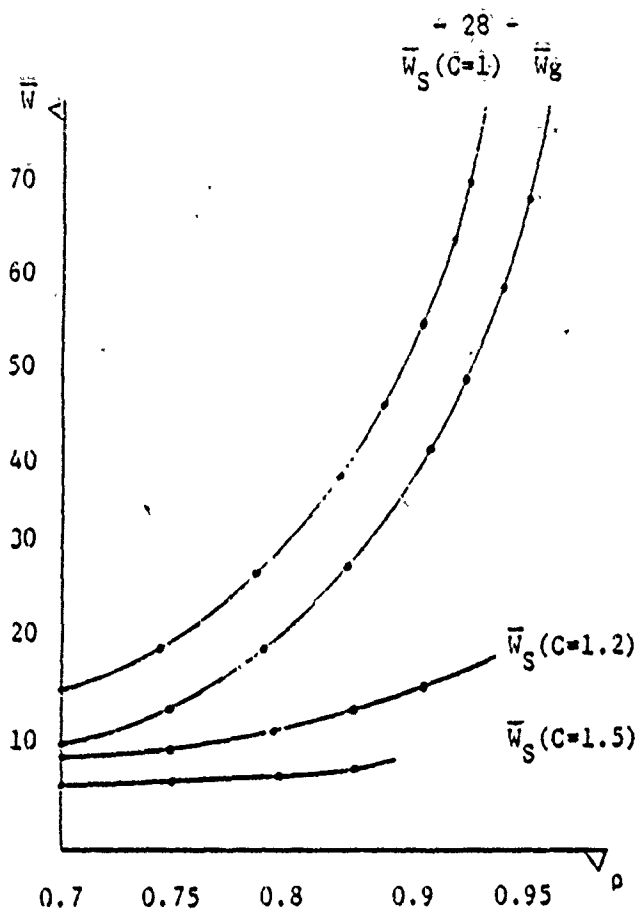


Figure 3  $\bar{W}_g$  and  $\bar{W}_S$   
 ( $r=3$ ,  $m=10$   
 independent tasks  
 $C_i = C$ ,  $m_i = \text{optimum}$ )

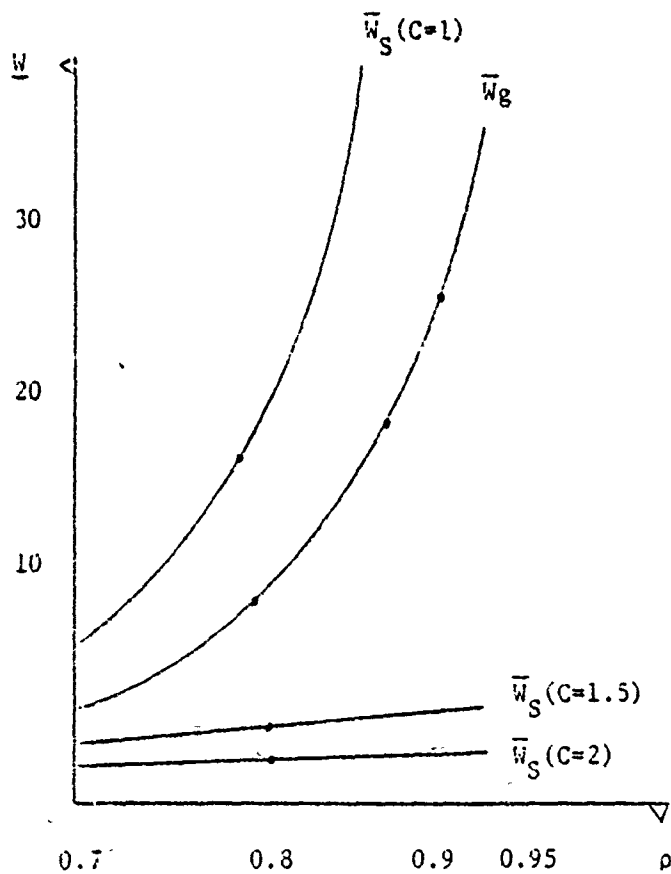


Figure 4  $\bar{W}_g$  and  $\bar{W}_S$   
 ( $r=m=3$   
 independent tasks)

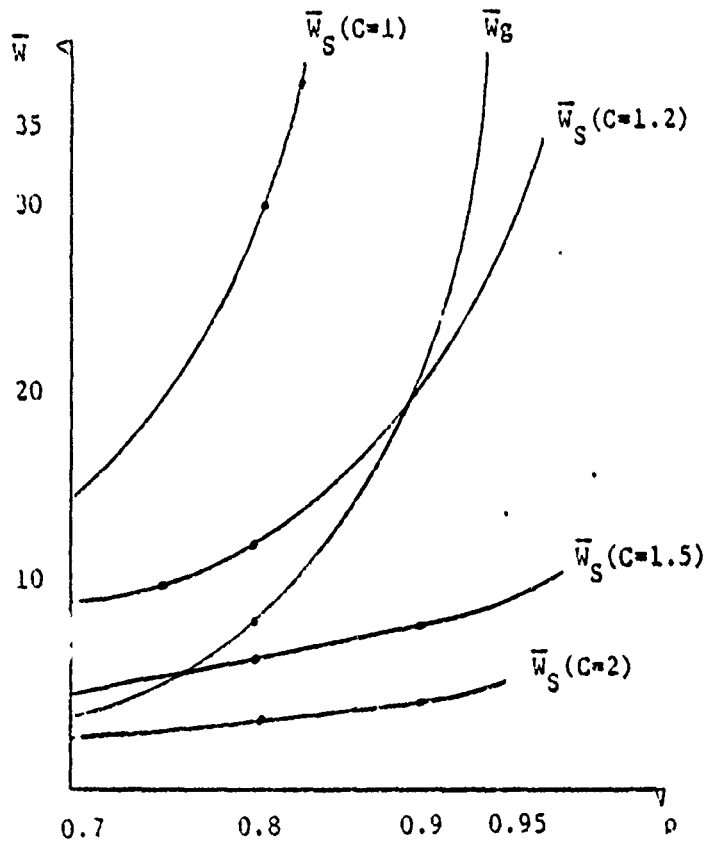
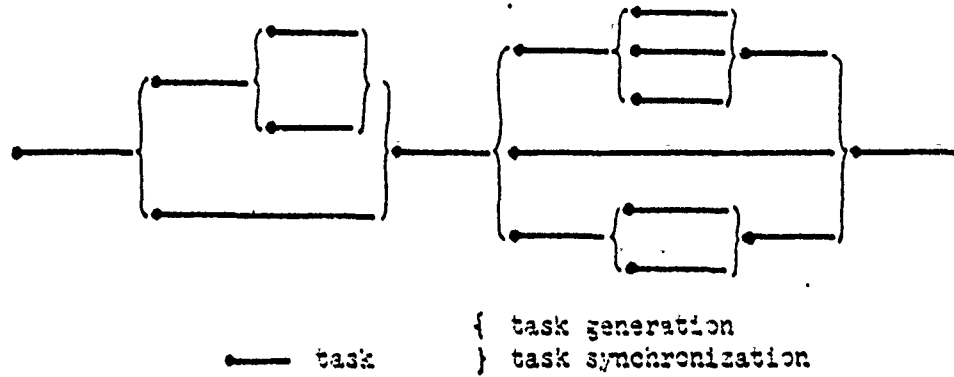
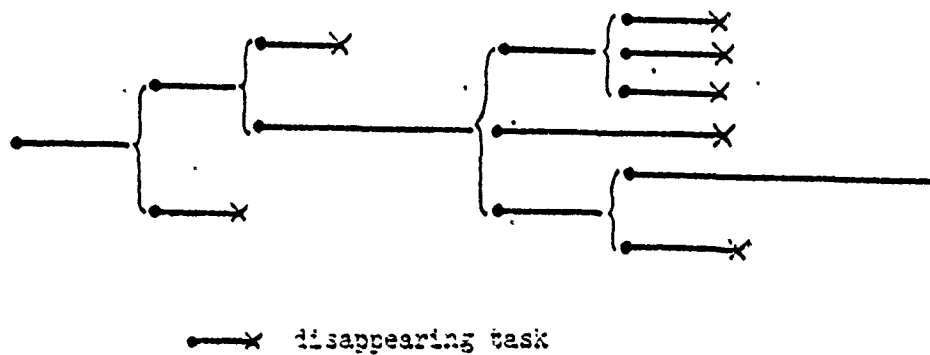


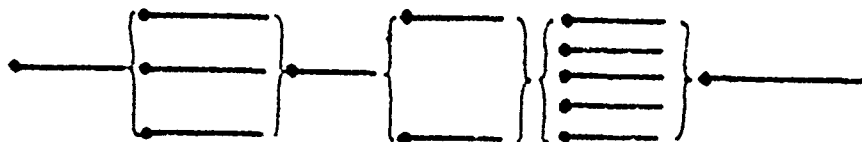
Figure 5  $\bar{W}_G$  and  $\bar{W}_S$  under Heavily Loaded Condition  
 $r = m = 3$  ,  $C_i = C$   
 special purpose processors pipelined



(a) A graphical model of a job

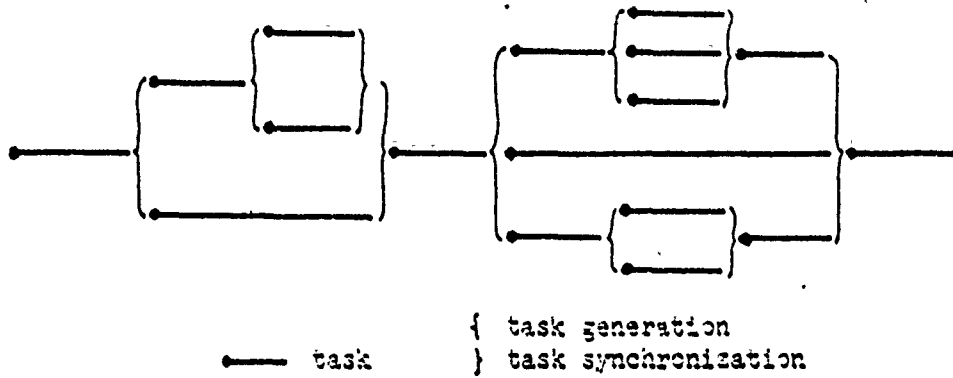


(b) No synchronization model of a job

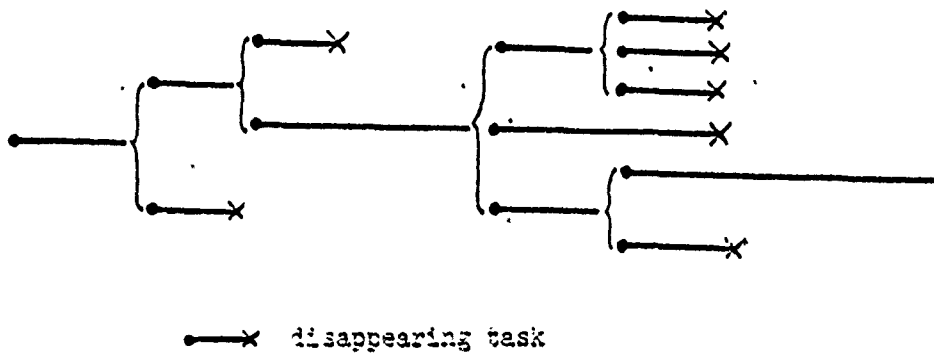


(c) Full synchronization model of a job

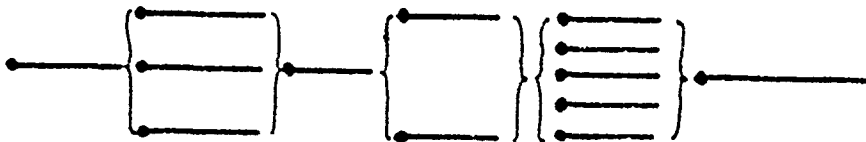
Figure 6 Models of Job Structure



(a) A graphical model of a job



(b) No synchronization model of a job



(c) Full synchronization model of a job

Figure 6 Models of Job Structure

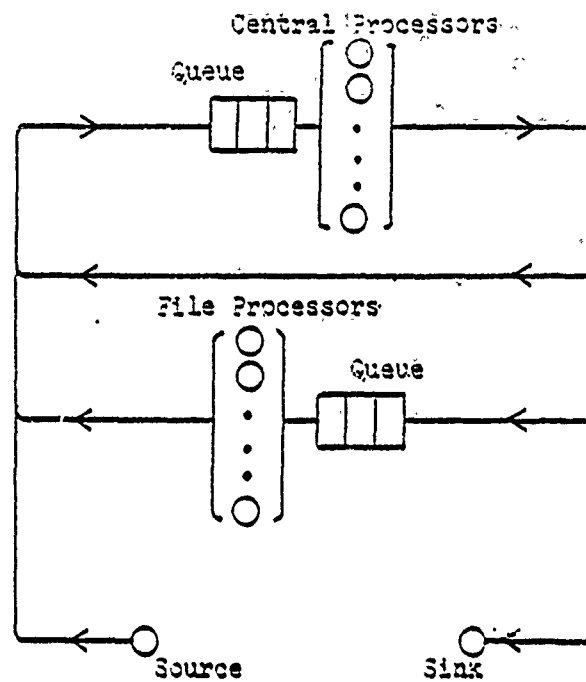
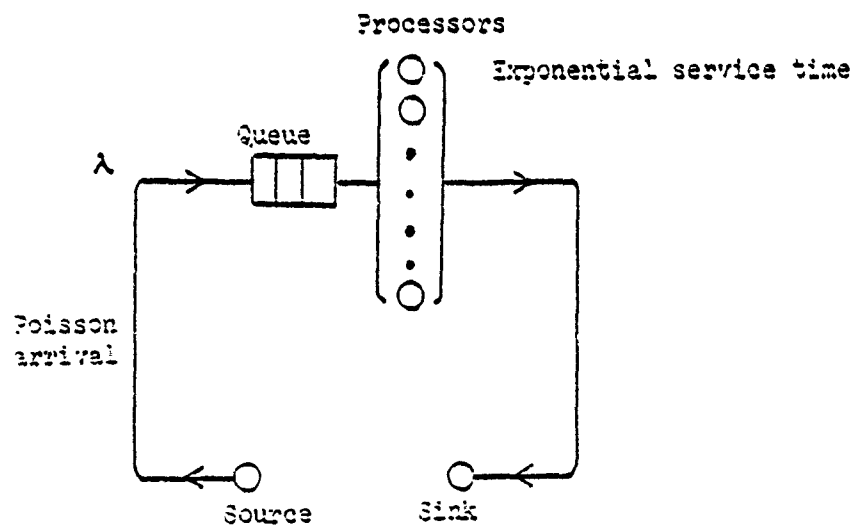
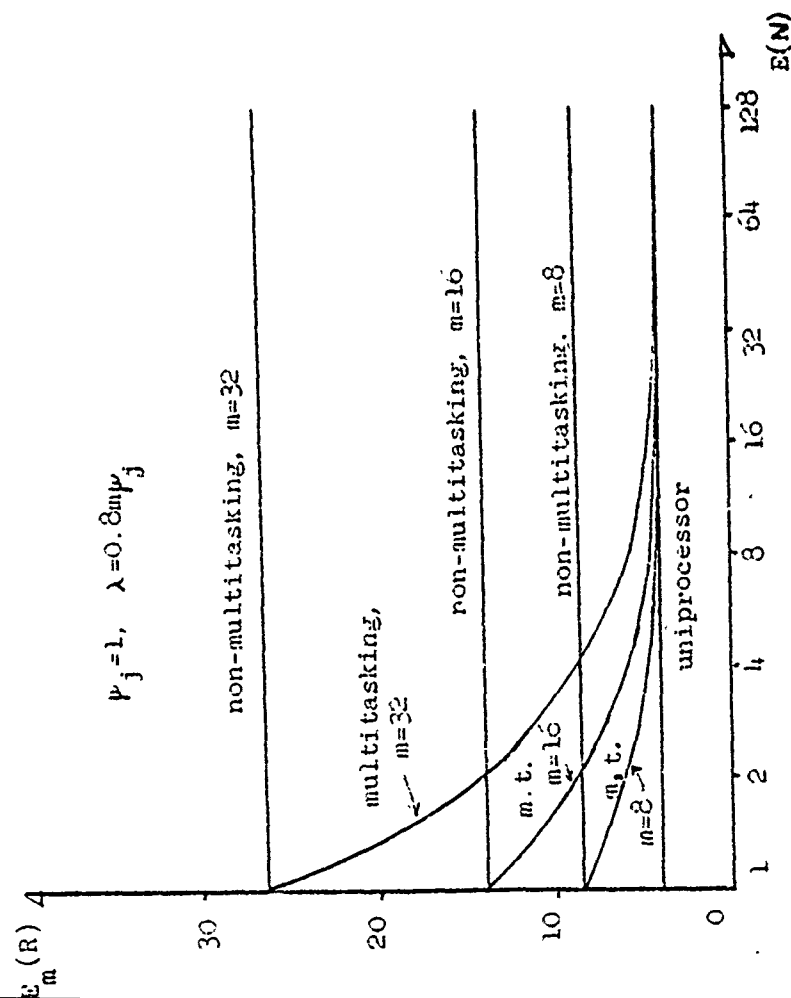


Figure 7. An Example of Open Queueing Network Model

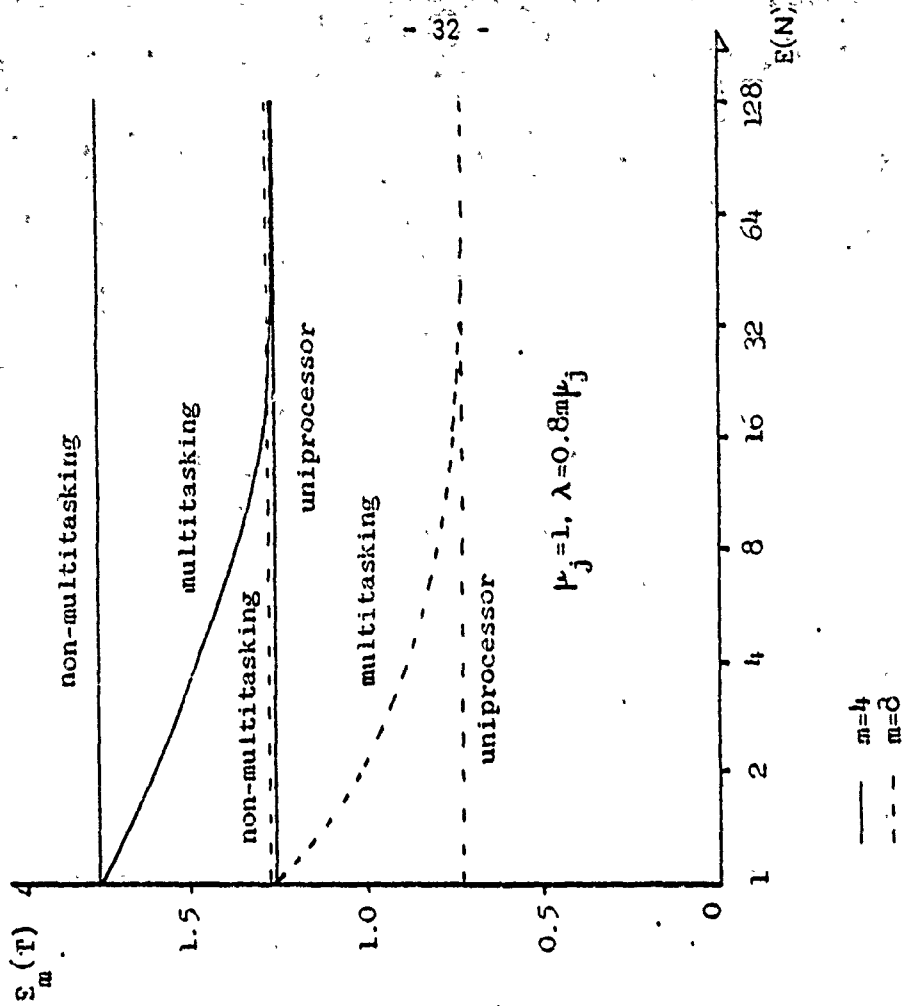


A job consists of a single stage of tasks

Figure 8. Multi-server Queueing System

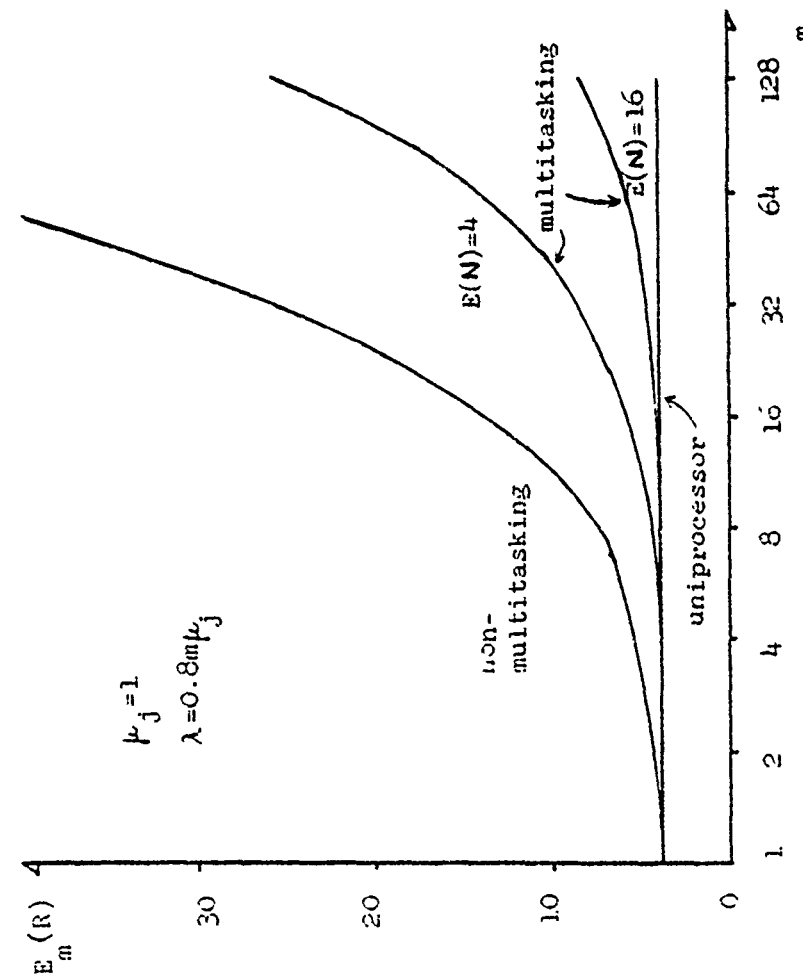


(a) The remaining service time

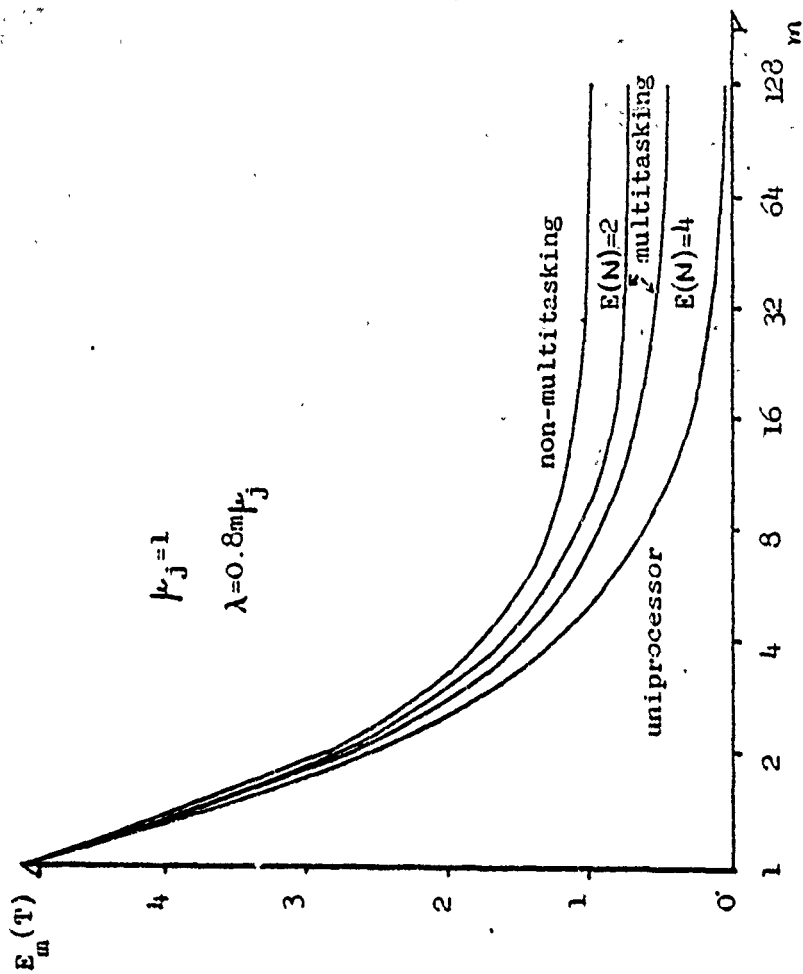


(b) The job turn around time

Figure 4 The effect of degree of concurrency on system performance



(a) The remaining service time



(b) The job turn around time

Figure 10 The effect of the no of processors